VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AGENTNÍ SYSTÉM PRO VYHLEDÁVÁNÍ NA REALITNÍCH SERVERECH

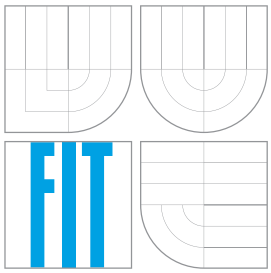
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ŠIMARA

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AGENTNÍ SYSTÉM PRO VYHLEDÁVÁNÍ NA REALITNÍCH SERVERECH

AGENT SYSTEM FOR SEARCHING ON REAL ESTATE SERVERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL ŠIMARA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAKUB FILÁK
SUPERVISOR

BRNO 2010

Abstrakt

Cílem této bakalářské práce je návrhnout a implementovat agentní systém pro vyhledávání na realitních serverech. Agentní systémy jsou v dnešní době jedním z populárních témat v oblasti umělé inteligence a nabízí velkou řadu možných využití. Jedním z nich je například vyhledávání informací v prostředí webu. V této práci jsou popsány některé existující vyhledávací systémy založené na agentním přístupu. Ty sloužily jako inspirace při navrhu výsledného systému, který se snaží vyhledat informace na realitních serverech a poskytnout jejich přehled.

Abstract

Object of this bachelor thesis is design and implement an agent system for searching on real estate servers. Nowadays are agent systems one of the popular topics in artifical intelligence and offers a wide range of possible uses. One of them is searching for information on the Web. In this paper are described existing agent–based retrieval systems. They serve as insipiration to design the final system, which search information on real estate servers and create an overview of them.

Klíčová slova

Agent, agentní systém, informační market, vyhledávací systém, realitní server, získávání dat.

Keywords

Agent, agent-based system, information market, searching system, real estate server, data mining.

Citace

Michal Šimara: Agentní systém pro vyhledávání
na realitních serverech, bakalářská práce, Brno, FIT VUT v Brně, 2010

Agentní systém pro vyhledávání na realitních serverech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jakuba Filáka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Šimara
13. května 2010

Poděkování

Chtěl bych poděkovat svému vedoucímu panu Ing. Jakubu Filákovi za cenné rady, nápady, doporučení a odbornou pomoc při vytváření této bakalářské práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Cílem této práce je navrhnout a implementovat agentní systém pro vyhledávání na realitních serverech. Vyhledávání je snadnější, pokud jsou věci uspořádané. Vzhledem k obrovskému množství informací na internetu a rychlosti, kterou přibývají, jsme nuceni vytvářet systémy, které by přinesly pořádek do tohoto chaosu [1].

Získávání informací z realitních serverů je mnohdy zdlouhavé a čas je pro většinu uživatelů velmi vzácný. Cílem tedy bude navrhnout takový systém, který by proces vyhledávání informací na realitních serverech co nejvíce usnadnil a který by také zároveň šetřil čas uživatelů strávený nad hledáním konkrétních informací.

Pro vyhledávání informací v prostředí Internetu byla navržena již řada systémů. Mezi nimi se nachází i zástupci multiagentních systémů, na které se tato práce hlavně zaměřuje. Ke známějším multiagentním systémům navrženým pro vyhledávání například patří SoftBotSearch a SAIRE. Tyto systémy jsou v této práci blíže popsány a na základě získaných informací zde byl navržen jednodušší model multiagentního systému určeného pro vyhledávání na realitních serverech.

Na začátku této práce jsou vysvětleny základní pojmy jako je agent a agentní systém. Následně jsou zde popsány příklady některých existujících systémů určených k vyhledávání, které jsou založeny na agentním přístupu. Dále jsou uvedeny základní architektury informačních marketů využívaných v agentních systémech. Následující část práce se zabývá samotným návrhem agentního systému pro vyhledávní na realitních serverech. Na začátku této části je popsána navržená funkcionalita celého systému, za kterou následuje popis řešení jednotlivých funkcí. Na kapitolu zabývající se návrhem pak navazuje kapitola, která popisuje samotnou implementaci celého systému. V ní jsou například uvedeny použité nástroje, jednotlivé části projektu a nechybí v ní ani detailní popis způsobu implementace agentů. V závěru se nachází zhodnocení této práce a popis možností dalšího vývoje navrženého systému.

Kapitola 2

Agent

Citace z [14]:

> Pojem agent vychází z latinského slova *agentum*, jehož význam je „ten, kdo jedná“. Stejného významu se dobereme, i pokud zkoumáme význam tohoto slova v českém jazyce. Agentem je obvykle chápána osoba, která jedná v zájmu jiné osoby, společnosti, nebo státního celku, tedy nějakého svého klienta. Předpokládá se, že agent přijal roli zastupovat klienta a jednat v jeho prospěch. Tím je koncepčně dána i funkce agenta v agentním systému.

V tomto textu se budeme zabývat umělými agenty, kteří mají podobné schopnosti jako agenti v lidské podobě. Umělí neboli programoví agenti jsou realizováni formou implementovaných kódů a algoritmů [13].

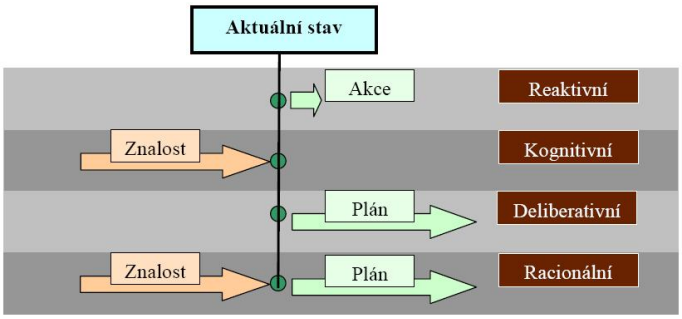
2.1 Základní typy agentů

Na základě různého chování můžeme rozlišit následující typy agentů [14]. Způsob činnosti některých z nich je znázorněna na obrázku 2.1.

- **Reaktivní agent:** Tento agent pracuje na principu zákona akce a reakce. Bez vnitřní reprezentace znalostí o prostředí, ve kterém se nachází, reaguje na jeho změny tak, aby dosáhl cíle, který mu byl určen. Jeho reakce nejsou výsledkem výpočtů či dedukcí na základě znalostí, ale pouze reakcemi na podněty. Tato jeho vlastnost se nazývá jako *reaktivita* a je vedle *autonomity* druhou významnou vlastností agentů [14].
- **Deliberativní agent:** Tento agent se od reaktivního agenta odlišuje svou schopností plánovat postup svých akcí za účelem dosažení cíle. Agent tedy musí mít schopnost různých výpočtů, které představují vnitřní činnost agenta. K dosažení svých záměrů pak agent ovlivňuje okolní prostředí tak, aby získal nějakou výhodu. Toto jednání je další z často uváděných vlastností agentů a nazývá se *proaktivita* [14].
- **Agent inteligentní:** Tento agent se snaží řešit jednotlivé úlohy za pomoci své „inteligence“. Ve většině případů se jedná o využití logické dedukce. Záměry agenta souvisí s účelem jeho vzniku, mohou se ale dynamicky měnit v průběhu jeho činnosti na základě informací získaných od jiných agentů [14].
- **Kognitivní agent:** Kognitivní agent má schopnost vyvozovat logické závěry ze svých pozorování okolního prostředí. Takový agent musí především být schopen se učit a

vytvářet si svou vlastní bázi znalostí. Do ní si během svého působení ukládá informace získané interakcí s okolím nebo znalosti získané dedukcí. Kognitivní agent nemusí mít nutně deliberativní schopnosti. Pak provádí pouze vnitřní akce, například analyzuje scénu, provádí překlad, nebo získává znalosti (dolování znalostí z dat) [14].

- **Racionální agent:** Racionální agent má všechny výše uvedené vlastnosti a jeho struktura obsahuje jak plánovací jednotku, tak i kognitivní jednotku včetně báze znalostí. Je to agent, který je na základě svých poznatků schopen se učit a pak plánovat svoji činnost tak, aby dosáhl svých cílů racionálním způsobem. Stojí nejvýše na pomyslné hierarchii uvedených agentů [14].



Obrázek 2.1: Rozdělení agentů [13]

2.2 Základní role agentů

Kromě jednoduchých agentních systémů existují i multiagentní systémy, ve kterých je každému agentu přiřazena nějaká role, kterou vykonává. Jeden agent může mít v jednom okamžiku přiřazeno i více rolí současně. Podle činnosti agenta pak můžeme rozlišit 3 základní typy agentů [6]:

1. **Koordinátor:**
   Koordinátor se obvykle vyskytuje na nejvyšší úrovni agentní hierarchie. Jeho úkolem je přidělovat agentům „pátračům“ požadavky, které přijal od uživatele. Následně pak vrací uživateli odpověď, kterou obdržel od pátračů.

2. **Pátrač:**
   Zástupci tohoto typu agenta vyhledávají informace na internetu, nebo v jiném prostředí. Podle způsobu získáváni je můžeme dále rozdělit na:

   - Mobilní agent – Mobilní agenti putují po internetu na servery, které prohledávají. Tyto servery by měly obsahovat informace odpovídající požadavkům uživatele.
   - Statický agent – Statičtí agenti se většinou specializují na jeden konkrétní vyhledávací nástroj a jemu posílají požadavky zadané koordinátorem.

3. **Učící se agent:**
   Tento agent zpracovává výsledky práce pátracích agentů a odezvu uživatele na tyto výsledky. Snaží se pochopit uživatelovy záměry a preference týkající se jeho požadavků.

Kapitola 3

Vyhledávání informací v prostředí webu

V dnešní době se pro vyhledávání informací na internetu velmi často využívá internetových vyhledávačů. Tyto vyhledávače využívají automatizované programy nazývané také jako „boty“. Boti procházejí stránky na internetu a postupně je indexují, díky čemuž umožní jejich snadné a rychlé vyhledání. Výsledky nalezené pomocí tohoto způsobu jsou následně řazeny podle hodnocení odkazovaných stránek, které je označováno také jako *page rank*. Tento způsob vyhledávání je využíván například jedním z nejpoužívanějších vyhledávačů, kterým je Google [2, 4]. Každý nalezený výsledek obsahuje odkaz na cílovou stránku a stručné informace o obsahu této stránky. V těchto informacích jsou obsažena klíčová slova spolu s kontextem, ve kterém byla použita. Díky tomu má uživatel možnost předem zjistit, jestli je nalezený výsledek relevantní k jeho požadavku. Bohužel ale někdy ani tyto informace nejsou dostačující k tomu, aby se dokázal rozhodnout o relevantnosti odkazovaných stránek, a musí jednotlivé stránky prohledat ručně.

Výše popsaný způsob vyhledávání ale nenabízí dostatečné možnosti specifikace požadavku. Z toho důvodu obsahuje většina realitních serverů vlastní vyhledávače, jejichž možnosti škálovatelnosti jsou v oblasti vyhledávání realit vyšší. Po nastavení kritérií do těchto vyhledavačů jsou zobrazeny nalezené výsledky, které splňují všechny zadané kritéria. Možnosti nastavení vyhledávání se ale server od serveru liší a uživatelé se musí neustále přizpůsobovat novým prostředím. Navíc jsou při vyhledávání nových informací uživateli neustále zobrazovány i výsledky, které již shlédl a které ho nezajímají, což prodlužuje celkovou dobu hledání.

3.1 Agentní vyhledávání

Agentní vyhledávání je, na rozdíl od běžného internetového, zaměřeno více na zpracovávání konkrétních požadavků uživatele a tomu jsou také přizpůsobeny architektury agentů. Agentní systémy mohou obsahovat agenty, kteří jsou schopni učit se. Tito agenti se pak v průběhu své činnosti zdokonalují a přispívají k vylepšování celého systému. Agenti mohou mít zároveň schopnost přizpůsobovat se změnám v jejich prostředí.

Agenti hledají informace v prostředích, pro které jsou specializovaní, což vede ke zlepšení kvality získávaných výsledků z jednotlivých zdrojů informací. Jednotliví agenti spolu mohou navzájem komunikovat a předávat si informace na místech označovaných jako *information markets*. Tyto místa budou v následujícím textu označována pomocí českého termínu jako

informační markety a budou popsány v kapitole 4. Komunikace mezi agenty umožňuje zefektivnění samotného procesu vyhledávání.

Většina agentních systémů je založena na víceúrovňovém multiagentním přístupu. V takovémto systému tvoří agenti určité hierarchické společenství, na jehož vrcholu se většinou nachází agent zvaný jako „koordinátor“. U vyhledávacích systémů je cílem tohoto agenta řízení činnosti agentů „pátračů“. Koordinátor předává pátračům dotazy, které obdržel od uživatele, a následně přijímá od pátračů odpovědi na jeho dotaz. Koordinátor může uživatelův dotaz rozšířit o informace, které pomůžou zefektivnit vyhledávání.

Existují dva základní typy agentů pátračů. Prvním typem jsou statičtí pátrači. Ti jsou často zaměřeni na jeden konkrétní vyhledávací nástroj, kterému posílají upravené požadavky od koordinátora. Tento způsob vyhledávání je znám pod pojmem „meta–vyhledávání“.

Druhým typem jsou mobilní agenti, kteří putují po Internetu na jednotlivé servery, které následně prohledávají. Po dokončení prohledávání se vrací zpět ke koordinátorovi, aby mu předali získané informace.

Agenti se pro vyhledávání hodí také díky své schopnosti učit se. V takovémto případě mluvíme o tzv. „učícím se agentovi“. Tito agenti pak mohou na základě získaných informací zlepšovat svoji činnost [6].

3.2 Existující řešení agentního vyhledávání

Abychom lépe porozuměli agentním a multiagentním systémům a možnostem, které nabízí k vyhledávání, jsou v této kapitole popsány dvě existující řešení těchto systémů. U každého z nich jsou uvedeny základní typy agentů, které se v něm vyskytují a které se podílejí na jeho činnosti. Součástí je také popis činnosti těchto systémů.

3.2.1 SoftBotSearch

SoftBotSearch poskytuje vyšší úroveň ve vyhledávání informací než např. Google, Yahoo a další vyhledávací systémy. Ty hledají jen na základě výskytu slov v dokumentech [6]. SoftBotSearch se oproti těmto vyhledávačům liší tím, že se navíc snaží tisíce nalezených dokumentů třídit do shluků a každému shluku přiřadit reprezentativní frázi, která nejlépe popisuje všechny dokumenty v něm [1].

Hlavní kroky vyhledávače při vyhledávání [6, 1]:

1. **Získání dokumentu**
   K získávání dokumentů jsou využívány vyhledávače jako Google, Yahoo a MSN Search. Z každého z nich je staženo prvních 100 odpovídajících dokumentů, a ty které jsou v HTML formátu, jsou dále zpracovávány. Z těchto dokumentů jsou ponechány jen důležité části jako nadpisy, klíčová slova, podnadpisy a vyznačené části, ty pak charakterizují typ dokumentu.

2. **Preprocessing**
   V tomto kroku dochází k odstranění slov, která se vyskytují v daném jazyce velmi často a nenesou žádnou významovou informaci. Tato slova jsou také označována jako *stopslova*. Jedná se například o spojky, předložky, zájmena a některá slovesa [5]. Zbylá slova jsou pak převáděna na jejich základní tvar, tento proces se pak nazývá jako *stemming*.

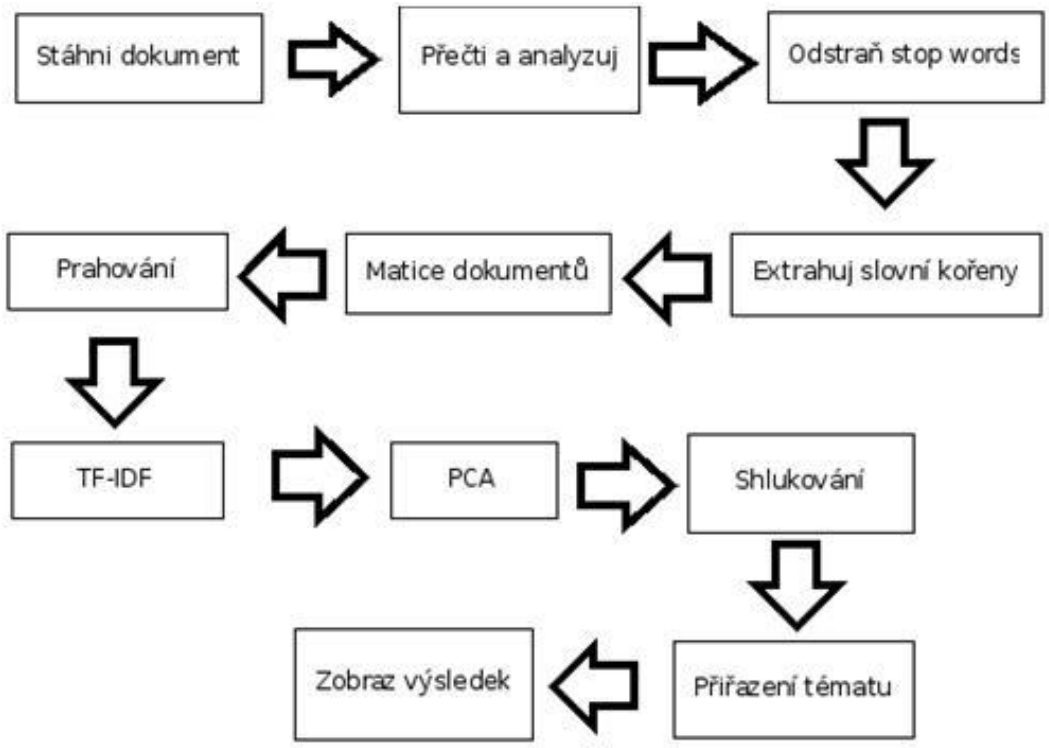
3. **Processing**
   V dalším kroku je vytvořena matice dokumentů. Slova vyskytující se v dokumentech velice zřídka nebo naopak velmi často jsou v této fázi odstraněny. Následně je frekvence jednotlivých slov převedena na TF-IDF[1] váhy , který ohodnocují význam slova v dokumentu [12]. Processing je zakončen analýzou hlavních komponent PCA[2], která se využívá ke snížení dimenze dat.

4. **Shlukování**
   K vytvoření nepřekrývajících skupin dokumentů je využito metody K-Means. Tato metoda je mezi algoritmy pro shlukování velmi populární hlavně kvůli své jednoduchosti a nízké výpočetní náročnosti.

5. **Extrakce témat**
   Z dokumentů ve skupině jsou vybrány fráze, které nejlépe danou skupinu popisují.

6. **Zobrazení výsledků**
   K zobrazení výsledků se využívá grafické formy, ve které je u každé skupiny zobrazen předmět, kterého se skupina týká. Po kliknutí na danou skupinu můžeme procházet jednotlivé dokumenty v ní.

Posloupnost jednotlivých kroků je zobrazena na obrázku 3.1.



Obrázek 3.1: SoftBotSearch – kroky vehledávání

[1] Term Frequency - Inverse Document Frequency
[2] Principal Component Analysis

3.2.2 SAIRE – A Scalable Agent-based Information Retrieval Engine [10]

SAIRE je systém s multiagentní architekturou vytvořený za podpory NASA za účelem poskytování dat o Zemi a o Vesmíru širší veřejnosti pomoci Internetu. Systém je zajímavý tím, že uživateli umožňuje komunikovat se systémem libovolným způsobem (textem, řečí, graficky, atd.).

V systému SAIRE se vyskytují 3 typy agentů:

1. Agenti uživatelského rozhraní (user interface agent)
2. Koordinující agenti zvaní také jako zprostředkovatelé (coordinator, faciliator, mediator)
3. Vyhledávací agenti (domain specialist agent)

SAIRE má 3 protokolové vrstvy. První vrstva zahrnuje komunikaci mezi agenty a jejich vzájemnou koordinaci. Druhou vrstvu využívají vyhledávací agenti ke komunikaci se zdroji dat. A poslední vrstvou je síťový komunikační protokol pro komunikaci s agenty sídlícími v různých uzlech Internetu.

Agenti v SAIRE jsou cílově orientované autonomní inteligentní moduly, vyvíjené za účelem získávání informací. Díky tomu může uživatel nebo agent delegovat úkolu na další agenty s minimálním dohledem. Z toho důvodu multiagentní systém nabízí řadu schopností včetně následujících:

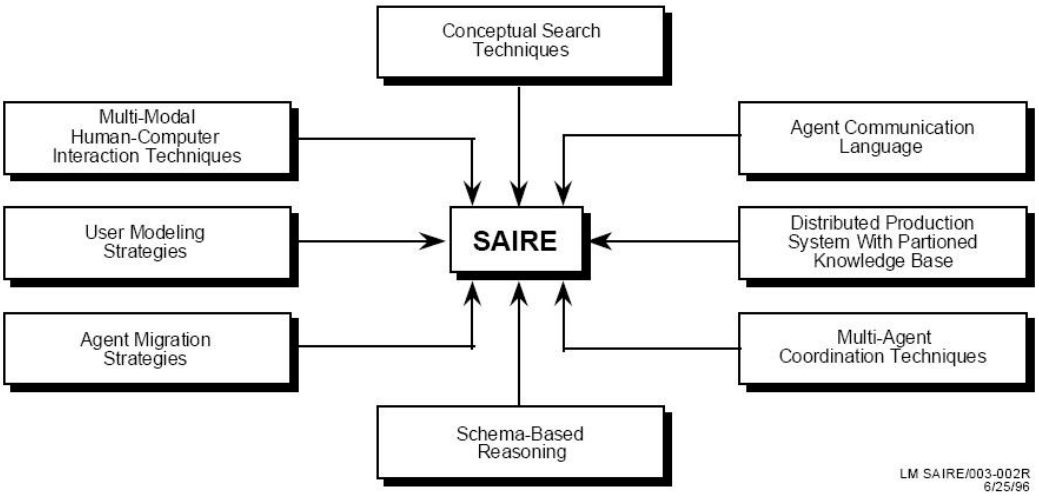
- Agenti mohou být nezávislí a mít nedeterministické chování. Každý agent je komplexní systém s vlastní soukromou bází znalostí.
- Agenti se mohou přemístit na jiné místo pro dosažení úkolu a tím snížit zatížení systému.
- Agenti smí být perzistentní a provádět dlouho trvající úkoly.
- Agenti spolu mohou navzájem komunikovat pomocí robustního agentního komunikačního jazyka, zatímco agenti uživatelského rozhraní, komunikují s uživatelem.
- Agenti jsou schopni se klonovat tak, aby zvládli složitější úkoly, které je potřeba provádět paralelně.

Mechanismus vyhledávání [6, 10]

V systému SAIRE je mechanismus vyhledávání založen na dvou hlavních úrovních. První úroveň se zabývá přiřazením témat k dotazu uživatele a v druhé úrovni dochází k analýze klíčových slov.

V prvním kroku se zjistí, jaká témata přísluší k uživatelskému požadavku. Toho je dosaženo vyhledáním klíčových slov z dotazu uživatele v matici témat. Matice témat obsahuje v jedné ose všechny známé výrazy a v druhé ose k nim přiřazená hlavní témata. Na základě klíčových slov z dotazu je vytvořen seznam témat, z něhož jsou vybrána ta nejvhodnější. Pokud systém nenalezne žádné téma, musí ho uživatel zadat manuálně výběrem z 13 dostupných témat.

Po vybrání témat přejdeme do další úrovně. V tomto kroku jsou do požadavku přidána další slova zvolená na základě jejich relevance ke klíčovým slovům a tématům zvoleným v prvním kroku. Výběr se provádí pomocí následujícího postupu. Ke každému klíčovému



Obrázek 3.2: SAIRE – schéma inovačních metod

slovu u každého specifického tématu existuje seznam relevantních termínů. Tyto termíny jsou ohodnoceny váhou podle toho, jak jsou relevantní. Váha termínu reprezentuje procentuální podíl dokumentů obsahujících klíčové slovo a všech přijatých dokumentů, který je normalizovaný číslem 5.

Vybrané relevantní termíny jsou k požadavku přidány podle toho, zda uživatel zadal více literálů do požadavku, čímž vytvořil méně konceptuální vyhledávání. K méně konceptuálním požadavkům jsou přidány více a méně vážené termíny. K více konceptuálním požadavkům jsou přidávány pouze ty termíny, které mají velkou váhu. Vybrané termíny jsou nakonec přidány do komunikačního rámce ACL[3] , který je dál zpracováván různými agenty. Agenti používají obsah rámce pro provádění dotazů nad databázemi.

Váhy vztahů termínů ke klíčovým slovům jsou přepočítány a aktualizovány při každém dotazu provedeném systémem na základě jeho výsledků. Tento mechanizmus udržuje sémantickou síť aktuální a také ji stále vylepšuje, čímž umožňuje poskytování stále lepších výsledků.

[3] Agent Communication Language

Kapitola 4

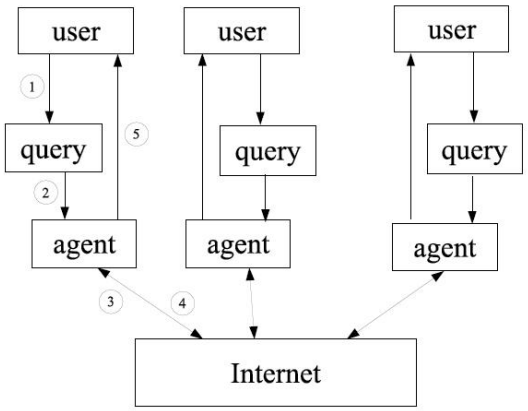
Informační market

V této části je blíže popsán informační market zmíněný již v předchozí části. Informační markety vznikly ve snaze zvýšit výkon vyhledávacích agentů a zjednodušeně se jedná o místa, na kterých se agenti shromažďují za účelem obchodování s nalezenými informacemi. Agenti zde ukládají jimi nalezené informace, za účelem jejich poskytnutí dalším agentům.

Architekturu agentního systému, který nevyužívá informačních marketů, nazýváme jako *no–exchenge*. Tato architektura je považována za standard a je využita jako základ pro složitější architektury využívající informačních marketů. Na základě zvoleného způsobu poskytování informací agentům můžeme rozlišit jejich dvě další architektury [6, 7].

4.1 No–exchange architektura

Cílem každého vyhledávacího agenta je získání informací z Internetu nebo jiného prostředí a předání jich uživateli. Tyto informace musí být samozřejmě dostatečně relevantní k jeho dotazu. U no–exchange architektury, na rozdíl od jiných zmíněných, nedochází k žádné výměně informací a vyhledávací agenti nemají možnost spolu navzájem komunikovat. Schéma aktivit agenta s touto architekturou je znázorněno na obrázku 4.1.



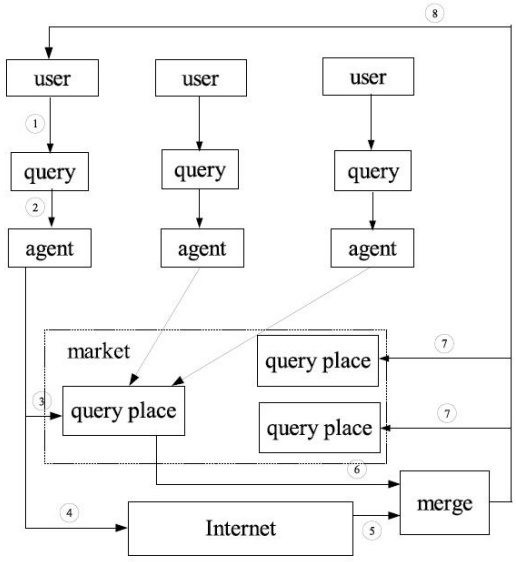
Obrázek 4.1: Informační market – no–exchange architektura

Uživatel podá dotaz (1), který je přiřazen některému agentovi (2). Agent prohledá Internet (3) a obdrži informace (4), které předá uživateli (5).

4.2 Free–exchange architektura

Jedná se o přímé rozšíření no–exchange architektury. Získávání dokumentů z Internetu stojí čas. Tato architektura se snaží snížit počet získávání stejných informací tak, že si je bude ukládat. Nejjednodušším způsobem, jak toho docílit, je ukládání si všech získaných dokumentů. Toho ale nemůžeme využít, kvůli jejich obrovskému množství.

Životní cyklus agenta ve free–exchange architektuře a v no–exchange architektuře je z velké části stejný. Jediným rozdílem je přidání informačního marketu, do kterého mohou agenti ukládat dokumenty ve prospěch ostatních agentů. Informační market obsahuje boty, kteří reprezentují agenty zpracovávané dotazy. Těmto botům je přiřazen list agentem nalezených dokumentů. Z toho důvodu je tento bot nazván jako „query place“.



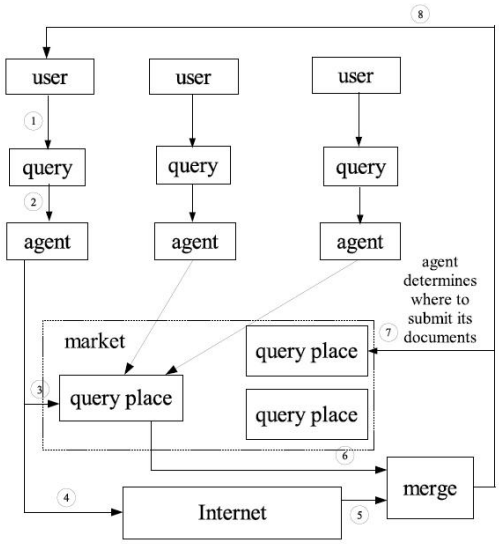
Obrázek 4.2: Informační market – free–exchange architektura

Obrázek 4.2 ilustruje free–exchange architekturu. Uživatel vytváří dotazy (1), které jsou přiřazeny některému dostupnému agentovi (2). Agent získá informace z marketu (3) a následně prohledá Internet (4). Získané informace z Internetu (5) a z marketu (6) pak sloučí do jednoho seznamu, který je před předáním uživateli (8) odeslán do marketu všem botům zvaným jako „query place“ (7).

4.3 Paid–exchange architektura

Tato architektura informačního marketu je podobná free–exchange architektuře. Došlo ale k několika změnám. Informační market využívá méně místa, to je ale využíváno více efektivně. Market je složen ze stejných botů, jejichž funkce je ale trochu odlišná. Nyní mají agenti virtuální peníze, za které si mohou kupovat informace. Peníze mohou naopak agenti získat prodejem těchto informací. Všechny tyto transakce jsou prováděny prostřednictvím zmíněných botů (query place).

Na obrázku 4.3 je zobrazena paid–exchange architektura informačního marketu. Uživatel vytváří dotazy (1), které jsou přiřazeny některému dostupnému agentovi (2). Agent zjistí, jaké informace se nachází v marketu (3) a následně prohledá Internet (4). Po získání



Obrázek 4.3: Informační market – paid–exchange architektura

informací z Internetu (5) se může agent rozhodnout koupit další informace z marketu (6). Následně dojde ke sloučení těchto informací do jednoho seznamu a agent se může rozhodnout prodat jim nalezené informace v marketu (7). Poté, co nabídne agent své informace na marketu, je sloučený seznam s informacemi předán uživateli. Následně může být agentovi přiřazen další dotaz.

K porovnání výkonu systémů, které jsou založeny na klasické architektuře, a systémů, které jsou postaveny na architektuře využívající informačního marketu, byla vyvinuta simulační architektura IRIS[1]. Měření pomocí ní ukázaly, že efektivita agentních systémů s informačním marketem se zvyšuje kvadraticky v závislosti na počtu vyhledávacích agentů [6, 7].

[1]Information Retrivial Intelligent System

Kapitola 5

Návrh systému

Na začátku návrhu tohoto systému bylo potřeba stanovit, co od něj bude očekáváno. To znamená definovat jednotlivé funkce systému. Následně budou zváženy možnosti realizace jednotlivých funkcí a zvoleny vhodné způsoby jejich implementace. Pokud by byla implementace některé funkce příliš časově náročná, bude potřeba zvážit, jestli je tato funkce pro celkovou funkčnost systému nezbytná a jestli je nutné ji implementovat v jejím plném rozsahu. Pokud dojde k tomu, že bude od některé funkce upuštěno, bude vynaložena alespoň snaha k tomu, aby mohla být tato funkce později snadno přidána.

V této části práce bude nejdříve popsána navržená funkcionalita systému a následně se bude zabývat bližším popisem jednotlivých navržených funkcí. Na závěr bude uveden popis návrhu agentů v systému a způsobu komunikace mezi nimi. Popis realizace jednotlivých funkcí je uveden v kapitole 6.

5.1 Funkcionalita systému

Základní funkcí systému je vyhledávání na realitních serverech. V kapitole 3.1 jsme se seznámili s agentním vyhledáváním a zde popsaný přístup bude uplatněn i při návrhu tohoto systému. Na první pohled bude možná systém připomínat realitní server. Uživatel bude do systému zadávat požadované parametry pro vyhledávání obdobným způsobem, jako je tomu u integrovaných vyhledávačů nebo filtrů realitních serverů. Systém ale nebude obsahovat databázi s vlastními realitami resp. inzeráty, ale tyto informace bude získávat z jiných podporovaných realitních serverů pomocí agentů. Tito agenti se pokusí požadované informace na realitních serverech vyhledat a předat je zpátky uživateli. Nalezené informace budou ještě před předáním uživateli ohodnoceny. Způsob hodnocení jednotlivých výsledků bude popsán níže v této kapitole.

Nalezené výsledky a nastavení vyhledávání se budou ukládat do databáze. Díky tomu uživatel nebude muset znovu zadávat všechny parametry, ale jen si zvolí jim již dříve vytvořené a nastavené vyhledávání a požádá systém o jeho aktualizaci. Ukládání dat do databáze je realizováno kvůli řadě funkcí, které by měly uživatelům vyhledávání na realitních serverech usnadnit a které využívají toho, že proces vyhledávání na realitních serverech má dlouhotrvající a periodický charakter.

Jak bylo zmíněno, jednotlivé nalezené výsledky budou hodnoceny a to nejen ze strany systému, ale také ze strany uživatele. Na základě získaných hodnocení bude systém řadit jednotlivé výsledky v jejich výsledném seznamu. Jednotlivé zmíněné způsoby ohodnocování budou popsány v kapitolách 5.2 a 5.3. Získané výsledky bude pak možné přiřazovat do

několika kategorií, které budou navrženy tak, aby přispěly k lepší orientaci mezi nalezenými výsledky.

Aby uživatel nemusel v případě nalezení malého množství informací zdlouhavě přenastavovat parametry vyhledávání ve snaze získat více výsledků, bude systém umožňovat vyhledání výsledků, které splňují jím zadané kritéria alespoň z určité procentuální části. Uživatel bude mít také možnost před vyhledáním podobných výsledků nastavit libovolné parametry jako povinné. Takto označené parametry pak bude muset splňovat každý nalezený výsledek. Jakým způsobem bude tato funkce řešena bude popsáno v sekci 5.5.

Při úvaze o zvýšení využitelnsti tohoto systému byla vzata v úvahu možnost návrhnutí systému tak, aby byl schopen vyhledávat i na jiných typech serverů. Z určitého hlediska lze totiž na realitní servery pohlížet jako na specifický typ inzertních serverů, který se od nich liší převážně svým zaměřením na konkrétní typ zboží. Stojí tedy za zvážení, jestli by určitá abstrakce problematiky nepřispěla k vyšším možnostem rozšiřitelnosti tohoto systému.

5.2 Hodnocení výsledků uživatelem

Jak bylo již v předchozí části zmíněno, jednotlivé nalezené výsledky budou ohodnocovány. Díky této klasifikaci výsledků se bude moci navrhovaný systém lépe přizpůsobovat požadavkům uživatelů. V rámci hodnocení výsledku dochází také k hodnocení realitního serveru, na kterém byl výsledek nalezen. Jednotlivé zobrazené výsledky jsou v tomto případě prostředníkem pro jejich hodnocení.

Hodnocení výsledků bude realizováno dvojím způsobem. Prvním z nich je přímé hodnocení. U každého tímto způsobem ještě neohodnoceného výsledku budou zobrazena 2 tlačítka. Kliknutím na první z nich bude zaznamenána do systému uživatelova spokojenost s nalezeným výsledkem a kliknutím na druhé bude zaznaménána jeho nespokojenost. Druhým zpusobem jakým bude uživatel hodnotit nalezené výsledky je nepřímé hodnocení. Nepřímé hodnocení bude získáváno na základě sledování činnosti uživatele. Systém bude sledovat, které výsledky si uživatel detailně prohlíží a do kterých kategorií si je zařazuje. Výsledky se budou moci vyskytovat ve 4 základních kategoriích:

- **Nové výsledky**
  V této kategorii se nachází výsledky při jejich prvním nalezení a zobrazení uživateli. Po kliknutí na odkaz, který otevře uživateli stránku, na které byl výsledek nalezen, je výsledek přesunut do kategorie standardních výsledků. Do této kategorie nebude uživatel moct výsledky znovu zařadit.

- **Standardní výsledky**
  V této kategorii se budou nacházet ty výsledky, které uživatel do žádné kategorie explicitně nepřiřadil. Budou zde také přesunuty výsledky, které byly před aktualizováním vyhledávání označeny jako nové .

- **Oblíbené výsledky**
  Pokud uživatele některý výsledek zaujme, může si jej zařadit do této kategorie, aby jej později snáze našel.

- **Skryté výsledky**
  Do této kategorie si uživatel bude moct naopak zařadit výsledky, které ho již nezajímají.

Systém bude ke každému nalezenému výsledku a realitnímu serveru uchovávat v databázi jeho hodnocení. Toto hodnocení bude obsahovat například počty kliknutí na výsledky resp. na výsledky nalezené na konkrétním serveru, nebo počet výsledků zařazených v kategorii oblíbených výsledků. Získané informace pak budou podkladem pro váhovou funkci 5.1 pro výpočet hodnocení. Jednotlivým parametrům bude přiřazena určitá váha, což by mělo umožnit následné doladění této funkce k větší spokojenosti uživatelů. Hodnocení výsledků a serverů bude jedním z parametrů na základě kterých budou řazeny výsledky v zobrazovaném seznamu.

$$h = \frac{v_{cc} \cdot cc + v_{cof} \cdot cof + v_{coh} \cdot coh + v_s \cdot s \ + v_{us} \cdot us}{(d+1) \cdot v_d} \tag{5.1}$$

V této funkci mají jednotlivé parametry následující význam:

- h značí výsledné hodnocení spočítané ohodnocovací funkcí,
- cc značí počet kliknutí na výsledek,
- cof značí počet výsledků zařezených v kategorii oblíbených výsledků,
- coh značí počet výsledků v kategorii skrytých výsledků,
- s značí počet výsledků se kterými jsou uživatelé spokojeni,
- us značí počet výsledků se kterými uživatelé nejsou spokojeni,
- d značí počet dní od přidání serveru do systému,
- v_x znači váhu pro parametr x.

5.3 Ohodnocení výsledků systémem

Jednotlivé nalezené výsledky budou systémem ohodnoceny na základě toho, do jaké míry splňují kritéria uživatele. Systém zjistí počet kritérií, které nalezený výsledek splňuje, a počet kritérií, které uživatel zadal. Podíl těchto hodnot pak představuje ohodnocení nalezeného výsledku. Vypočítané hodnocení je dále normalizováno číslem 100 a zobrazeno u výsledku jako procentuální část, ze které výsledek splňuje uživatelem zadané kritéria.

Na základě tohoto hodnocení bude uživatel moci výsledky filtrovat ve výsledném seznamu. Nastavením spodní hranice hodnocení na 80 % reprezentuje uživatelovo slevení z nároků na splění všech kritérií o 20 %.

5.4 Systematické řazení výsledků

Systematické řazení výsledků je jedna z funkcí, která by měla umožnit zobrazení kvalitnějších výsledků na začátku seznamu. Kvalita může být chápána a měřena různým způsobem, v tomto případě budou brány za kvalitnější výsledky, ty s lepším získaným hodnocením od uživatelů a ty které splňují nejvíce uživatelových kritérií. Díky tomu by se měl uživatel dostat k výsledkům, které zajímají i ostatní uživatele, na začátku svého vyhledávání. Je pravděpodobné, že by ho tyto výsledky mohly také zajímat.

Výsledky tedy budou řazeny na základě dvou dříve popsaných hodnocení. Prvním je vypočítané ohodnocení výsledku systémem, které bylo popsáno v sekci 5.3. Toto hodnocení označuje do jaké míry splňuje výsledek požadavky uživatele. Uživatel pravděpodobně bude chtít jako první v seznamu vidět výsledky, které splňují větší počet jeho kritérií, proto je toto hodnocení bráno jako hlavní kritérium pro řazení výsledků.

Druhým hodnocením je kvalita realitního serveru, na kterém byl výsledek nalezen, a výsledku samotného. Způsob získání tohoto hodnocení byl popsaný v kapitole 5.2. Důvodem použití tohoto hodnocení je snaha zajistit lepší umístění kvalitních výsledku, a také aby realitní servery, které poskytují více informací k nabízenému zboží byly zvýhodněny a tedy jejích výsledky umístěny na lepších pozicích v seznamu.

5.5 Hledání podobných výsledků

Hledání na realitních serverech je většinou zdlouhavý proces, při kterém se ne vždy podaří najít výsledky, které by splňovaly všechna zadaná kritéria. Pokud je nalezeno malé množství výsledků a žádný z nich nesplňuje uživatelovy představy, umožňuje mu systém vyhledat podobné výsledky, které splňují uživatelova kritéria alespoň z určené části. Za normální situace by byl uživatel nucen ručně přefiltrovat zadaná kriteria a doufat, že poté najde více vysledků. V takovém případě se může proces vyhledávání značně protáhnout a tomu se snaží tento systém zabránit. Proto systém implementuje funkci, která by měla dohledávání podobných výsledků značně usnadnit.

Výše zmíněné funkce bude docíleno postupnou exkluzí jednotlivých kritérií z požadavku uživatele. Mohou být vyloučena ale jen ta kritéria, která nejsou označena jako povinná. Je-li požadavek uživatele reprezentován jako množina kritérií, pak pro získání submnožin těchto kritérií, které splňují uživatelův požadavek z určité předem stanovené části musíme získat jednotlivé kombinace těchto kritérií. Nesmíme zapomenout na to, že povinná kritéria budou muset být obsažena ve všech těchto kombinacích. Dal-li se původní počet kombinací vyjádřit kombinačním číslem [3]

$$(k) = \begin{cases} \frac{n!}{k!(n-k)!} & \text{pro } 0 \leq k \ \leq n \\ 0 & \text{pro } k \ < 0 \text{ nebo } k > n, \end{cases} \tag{5.2}$$

kde n i k bylo rovno počtu kritérií v požadavku a výsledek byl tedy roven 1, pak snížením řádu, redukujeme počet kritérií v jednotlivých kombinacích a tedy i nároky na požadované výsledky. Upravený vzorec vyjadřující počet všech možných kombinací s ohledem na povinná kritéria je znázorněn níže.

$$k_2 = \lfloor min \cdot k \ + 1 \rfloor \tag{5.3}$$

$$\binom{n-p}{k_2-p} = \begin{cases} \frac{(n-p)!}{(k_2-p)!(n-k_2)!} & \text{pro } p \leq k_2 \leq n \\ 0 & \text{pro } k_2 < p \text{ nebo } k_2 > n \end{cases} \tag{5.4}$$

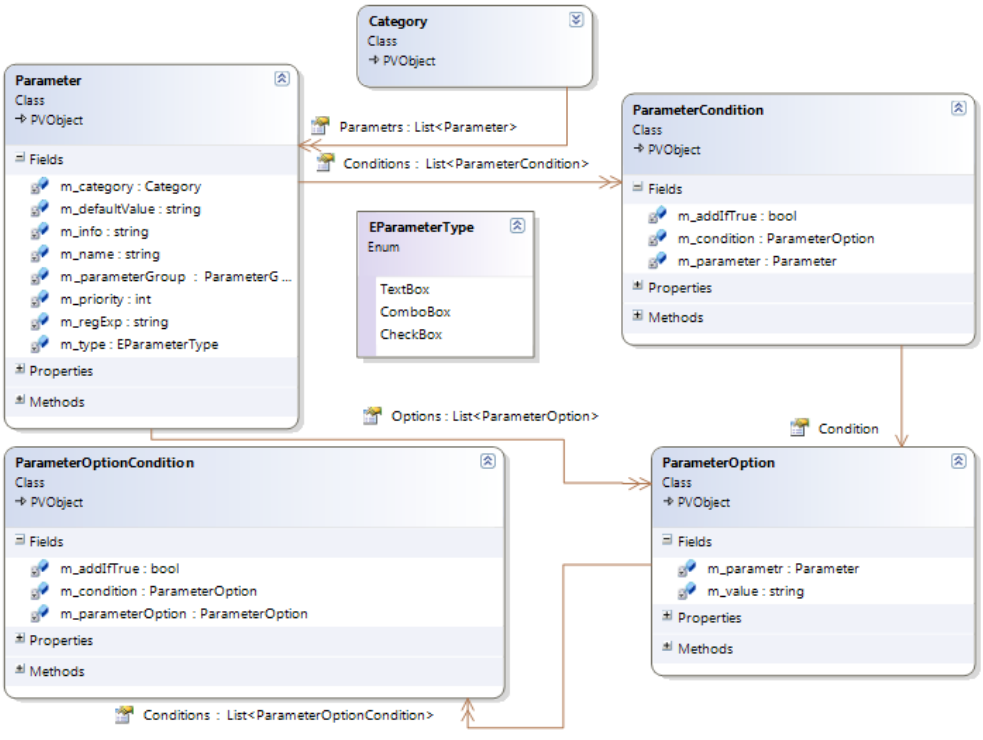
V tomto vzorci k_2 znázorňuje upravený řád kombinačního čísla tak, aby bylo zajištěno splnění minimálních požadavků uživatele. Parametr min zastupuje poměr počtu kritérií, která musí být splněna a počtu všech kritérií. Tento poměr si určuje uživatel a jeho defaultní hodnota je 1. Parametr p značí počet povinných kritérií. Předáním těchto kombinací kritérií agentům získáme všechny výsledky, které do určité míry splňují požadavky uživatele.

5.6 Parametry vyhledávání

Možnosti nastavení integrovaných vyhledváčů, popřípadě filtrů jednotlivých realitních serverů, se mohou výrazně lišit. Objevuje se tady problém, jak všechny možnosti těchto vyhledávačů a filtrů sjednotit a zahrnout do jednoho vyhledávače. Zároveň by měl systém počítat i s novými možnostmi, které se mohou vyskytnout při přidávání agenta pro vyhledávání na zatím nepodporovaném realitním serveru. To znamená, že je zapotřebí, aby systém umožňoval snadné přidávání nových parametrů pro vyhledávání. Pro získávání hodnot jednotlivých parametrů bývají nejčastěji využity následující ovládací prvky:

- TextBox – editovatelné textové pole
- CheckBox – zaškrtávací pole
- ComboBox – rozbalovací seznam

Často je možné narazit i na ovládací prvek označovaný jako RadioButtonList[1], ale jeho funkčnost se dá snadno nahradit pomocí prvku ComboBox. U ovládacího prvku TextBox se dále můžeme setkat s nejrůznějšími omezeními ve formě regulárních výrazů. Ty pak zajišťují, aby byla do prvku zapsána hodnota v požadovaném tvaru. Zřídka se pak vyskytují i prvky typu ComboBox, u nichž je zobrazení některých možností podmíněné nastavenou hodnotou v jiném prvku. Zároveň je zapotřebí dokázat rozlišit, jestli je parameter nastaven na výchozí hodnotu a jestli tato hodnota nebyla uživatelem změněna. Pokud jsou tyto informace vzaty v úvahu při návrhu výsledného systému, je možné sestavit následující diagram tříd.



Obrázek 5.1: Parametry - diagram tříd

[1]Seznam přepínacích tlačítek

Na diagramu 5.1 je zobrazeno 5 tříd a jeden výčtový typ. Nejdůležitější třídou na tomto diagramu je třída `Parameter`. Tato třída v sobě zahrnuje název parametru, detailnější informace, regulární výraz a typ parametru reprezentovaný již zmíněnou enumerací. Objekty této třídy jsou pak agregovány na třídu `Category`, jejíž přítomnost bude vysvětlena v sekci 5.7. Jednotlivé parametry pak mohou být podmíněny nastavením, nebo naopak nenastavením určité hodnoty v jiném parametru. Z tohoto důvodu je v návrhu třída `ParameterCondition`. Parametry získávané pomocí ovládacího prvku ComboBox budou nabízet na výběr jednotlivé možnosti, které reprezentuje třída `ParameterOption`. Jednotlivé možnosti mohou být ještě podmíněny hodnotou v jiném ovládacím prvku, pro tento případ je v návrhu třída `ParameterOptionCondition`.

5.7 Kategorie

Pokud budeme hledat na internetu nějaké realitní servery, určitě dříve nebo později narazíme i na inzertní servery. Na inzertních serverech většinou najdeme inzeráty rozřazené do různých kategorií, popřípadě je server zaměřen jen na jednu určitou kategorii. Pokud se nad tím zamyslíme, tak zjistíme, že inzertní servery se od těch realitních příliš neliší a jsou si v mnoha ohledech velmi podobné. Vezmeme-li toto v úvahu při návrhu našeho systému a zavedeme v něm kategorie, pak získáme do budoucna mnohem více možností jak náš systém rozšiřovat. Na jednotlivé kategorie se pak budou vázat různé parametry, které budou moci být při vyhledávání nastaveny.

Kvalita jednotlivých kategorií na inzertním serveru může být velmi odlišná. Nebylo by tedy úplně vhodným řešením, aby uživatelé hodnotili server jako celek. Praktičtější bude hodnocení jeho jednotlivých kategorií. Díky tomu se zbavíme nežádoucích vlivů způsobených rozdílnou kvalitou kategorií na jednom serveru a jednotlivá hodnocení pak budou více objektivní.

5.8 Návrh agentního systému

V kapitole 2.2 byly uvedeny základní role, které jsou ve většině vyhledavacích systémů agentům přiřazeny. S využitím těchto informací je možné navrhnout tento systém tak, že se v něm budou nacházet 2 základní typy agentů. Tito agenti se budou různým způsobem podílet na vyhledávání informací na realitních serverech. Tento systém je tedy navržen jako multiagentní systém (MAS), ve kterém spolu jednotliví agenti spolupracují. Součásti systému bude také agentní prostředí, do kterého se budou agenti hledači registrovat, a ve kterém budou přijímat příchozí dotazy. Toto prostředí bude poskytovat informace o těchto agentech, na základě kterých bude možné ovlivňovat činnost systému.

Základní typy agentů v systému:

1. **Hledač**
   Úkolem agenta hledače je prohledání realitního serveru, pro který byl navržen a nalezení všech výsledků, které splňují kritéria uživatele. První činnost tohoto agenta je registrace do agentního prostředí, následně se pak snaží z tohoto prostředí získávat dotazy, které jsou pro něj určeny a pokud nějaký získá zahájí se proces vyhledávání informací, v opačném případě může být agent uspán, nebo může provádět činnost, která mu do budoucna usnadní vyhledávání nových výsledků. Aby agent zbytěčně nehledal výsledky, které již byly nalezeny dříve, jeho prvním krokem v průběhu získávání odpovědi na dotaz bude cesta na informační market, kde se pokusí najít výsledky,

které splňují jemu předaná kritéria. Cestu na informační market si agent zopakuje ještě jednou, poté co prohledá realitní server a předá nalezené výsledky koordinátorovi. Podruhé zde ale půjde již za účelem poskytnutí jim nalezených výsledků.

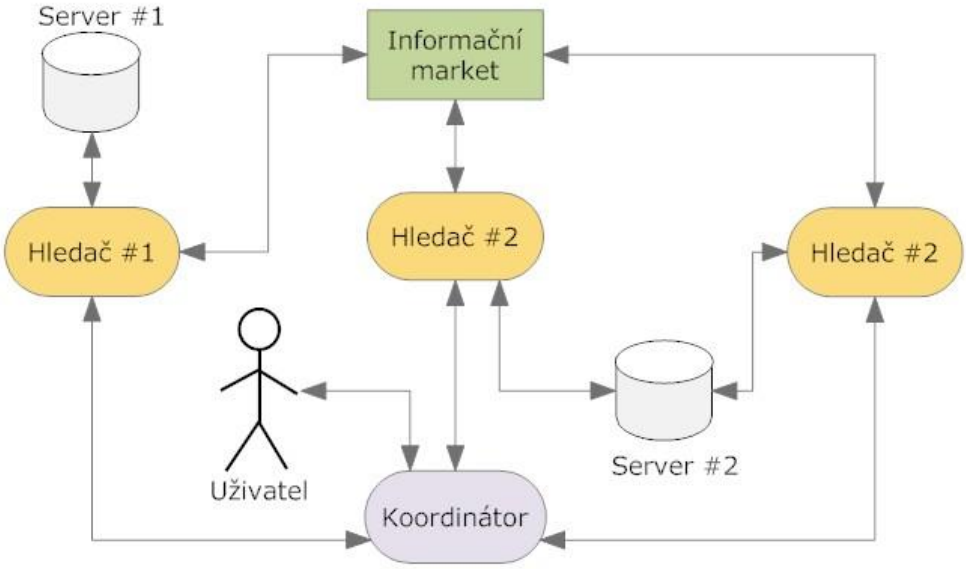
Vyhledávání informací na realitních serverech může být agentem hledačem prováděno různými způsoby. Hledač při tom může využívat regulárních výrazů, XSLT transformací a vytvořeného slovníku, ve kterém budou uloženy pro konkrétní parametry a agenty potřebné hodnoty. Tyto hodnoty pak může agent použít například k sestavení PHP dotazu v URL adrese.

Vytvoření agentů, kteří by zvládli samostatně vyhledávat realitní servery na internetu a následně se naučili z nich získávat požadované informace by bylo příliš komplikované. Zvoleným řešením je tedy vytváření agentů, kteří budou schopni vyhledávat na jednom konkrétním serveru, popřípadě na více serverech, jejichž struktura a další vlastnosti jsou podobné.

2. **Koordinátor**
   Jak již vyplývá z názvu agenta, jeho funkcí bude koordinovat činnost všech ostatních agentů. Tento agent bude komunikovat jak s uživatelem tak s agenty. Jeho úkolem bude také sledovat činnost uživatelů a zpracovávat jejich hodnocení jednotlivých výsledků. Z informací získaných od uživatelů se bude agent učit lépe řadit nalezené výsledky v seznamu předávaném uživatelům. Aby byly nalezeny i podobné výsledky, jak je uvedeno v kapitole 5.5, bude koordinátor od uživatele přijímat v dotazu parametr, na základě kterého se budou dát vytvořit dotazy předávané agentům hledačům. Každý takovýto dotaz bude reprezentován submnožinou zadaných kritérií. Tato submnožina bude muset obsahovat všechna kritéria, která byla označena jako povinná, a jistý počet nepovinných kritérií zjištěný z předaného parametru. U výsledků nalezených na základě těchto dotazů bude zajištěno, že nedojde k zanedbání většího poměru kritérií než uživatel požaduje. Koordinátor pak vytvořené dotazy předá agentům hledačům. Agent koordinátor bude mít také možnost sledovat rychlost jednotlivých agentů. V případě výskytu výraznější pomalejšího agenta hledače bude mít koordinátor možnost vytvořit dalšího hledače, který bude vykonávat stejnou činnost, a urychlí tak získávání informací z konkrétního realitního serveru. Díky tomu by měla být průměrná rychlost získávání informací z jednotlivých serverů vyrovnaná.

V předchozí části byl zmíněný pojem „informační market“. Účelem informačního marketu v systému bude umožnění výměny informací mezi agenty hledači. Výsledkům v marketu bude přiřazena určitá trvanlivost. To znamená, že se zde budou moci vyskytovat jen určitou omezenou dobu, než budou zahozeny. Tato trvanlivost může být ale prodloužena přidáním stejného výsledku s odlišnými kritérii. Počet kritérií je ale konečný a můžeme tedy s jistotou říct, že každý výsledek, který se zde nachází bude po určité době smazán. Tento systém má jednak urychlit činnost systému a jednak částečně zabránit poskytování výsledků, které se již nemusí na realitních serverech nacházet. Systém s tímto informačním marketem bude založen na free–exchange architektuře, která byla popsána v kapitole 4.2.

Na obrázku 5.2 je znázorněno, které prvky spolu budou navzájem komunikovat. Je zde možné vidět komunikaci mezi koordinátorem a uživatelem. Při této komunikaci bude docházet k předání uživatelova požadavku koordinátorovi a následovně k předání nalezených výsledků zpět uživateli. Podobně tomu je mezi koordinátorem a agenty hledači, pouze dojde k obměně rolí a koordinátor se bude nyní dotazovat agentů hledačů. Znázorněna je také výměnu dat mezi agenty hledači prostřednictvím informačního marketu. Poslední,



Obrázek 5.2: Schéma komunikace [8]

ještě nezmíněnou komunikací, je komunikace mezi agenty a realitními servery, ta již bude probíhat na aplikační vrstvě pomocí protokolu HTTP. V rámci této komunikace budou hledači prohledávat realitní servery a získávat od nich požadované výsledky.

Kapitola 6

Implementace

V této části je popsána samotná implementace agentního systému pro vyhledávání na realitních serverech. Na začátku této kapitoly budou uvedeny zvolené nástroje pro implementaci. Následovat bude popis vícevrstvé architektury a způsobu jakým je implementována datová vrstva. Další kapitola se bude zabývat způsobem implementace agentů a informačního marketu. V závěru této práce budou uvedeny funkce, od kterých bylo při implementaci kvůli časovému omezení upuštěno nebo nebyly implementovány v jejich plánovaném rozsahu.

6.1 Zvolené nástroje

Pro implementaci systému byl použit jazyk C# s platformou .NET Framework 3.5. Tento jazyk byl vyvinut firmou Microsoft a představuje ucelený nástroj vhodný jak pro tvorbu databázových programů, tak pro tvorbu webových aplikací. To je umožněno díky technologiím ADO.NET a ASP.NET, jenž jsou součástí .NET Frameworku [9].

K implementaci tohoto projektu bylo pak využito vývojového prostředí Microsoft Visual Studio 2008. Toto vývojové prostředí v sobě zahrnuje nástroje jako jsou editor kódu, debugger, designer a mnoho dalších. Při návrhu objektového modelu bylo využito funkce Visual Studia 2008, která umožňuje tvorbu diagramu tříd. V tomto diagramu jsou všechny objekty provázány s kódem, díky čemuž je diagram stále aktuální i při dodatečných úpravách. Pro ukládáni dat jsem zvolil možnost využití relačního databázového systému Microsoft SQL Server.

6.2 Vícevrstvá architektura

Tento systém je založen na vícevrstvé archtektuře, která se skládá ze tří standardních vrstev a jedné nestandardní:

1. **Datová vrstva**
2. **Aplikační vrstva**
3. **Prezentační vrstva**
4. **Vrstva zdrojů** – tato vrstva slouží pro ukládání například textových řetězců, které jsou zobrazeny jako chybová hlášení.

Celý systém je pak složen ze dvou hlavních částí. První se zabývá ukládáním dat do databáze a tvoří tedy převážně datovou vrstvu. Druhá část se již zabývá implementací samotného agentního systému pro vyhledávání na realitních serverech a je tvořena převážně aplikační a prezentační vrstvou. Systém je tedy složen ze dvou projektů. V prvním projektu je vytvořen objekt nazvaný jako PVObject, který zajišťuje validaci odvozených objektů a jejich ukládání do databáze. V druhém projektu se nachází samotný agentní systém, který je již abstrahovaný od datové vrstvy.

6.3 Ukládání perzistentních dat

Jak bylo zmíněno v předchozí kapitole 6.2 datová vrstva je řešena jako samostatná část projektu. V této vrstvě je řešeno ukládání perzistentních objektů do databáze a jejich opětovné načtení z databáze. Před uložením objektu do databáze musí samozřejmě dojít k jeho validaci, ta je řešena rovněž v rámci této části projektu. Výsledkem bude třída s názvem `PVObject`, která bude předkem všech perzistentních objektů v systému. Všechny objekty odvozené od této třídy pak budou poskytovat metody `Save()`, `Delete()` a `Validate()`. Pomocí statické třídy `Utility` pak bude možné takto uložené objekty z databáze znovu načíst. K tomu byly vytvořeny v této třídě statické generické metody `GetAll<T>()`, `GetAll<T>(string criteria)`, `GetObjectById<T>(int id)`.

Aby bylo možné takto s objekty pracovat, bylo zapotřebí využit reflexe a metadat ve formě atributů, které obsahují název tabulky a názvy sloupců, do kterých se budou zapisovat jednotlivé perzistentní hodnoty objektů konkrétní třídy. Důležitá je také třída `SqlHelper`, která obsahuje metody pro generování základních parametrizovaných příkazů, díky tomu by mělo být zabráněno útokům v podobě SQL injektáže [9]. Pomocí této třídy se také bude nastavovat `connection string`[1] pro připojení k databázi.

Objekt nabízí rovněž validaci objektu a to jak před zápisem hodnoty pomocí vytvořených vlastnosti, tak před samotným uložením objektu do databáze. Toho bylo dosaženo také díky reflexi a informacím předaným v atributech. Do atributů mohou být uloženy pouze předem známé hodnoty, nelze odtud volat žádnou metodu, pomocí které bychom určili určitá předem neznámá omezení. Z toho důvodu byla implementována možnost označit atributem „speciální“ metodu pro validaci. Tato metoda je pak volána při validaci prvku označeného v atributu u vytvořené metody. Pomocí atributu lze ale nastavit základní a nejčastěji využívané omezení:

- **NotNull**
  Pokud je nastaven tento parametr na true, pak hodnota nesmí být při validaci rovna defaultní hodnotě. Pokud se jedna o parametr typu string je provedena i kontrola jestli není řetězec prázdný.
- **Unique**
  Nastavením tohoto parametru zajistíte jedinečnost hodnot v danném sloupci tabulky.
- **MaxLength**
  Pomocí tohoto parametru lze kontrolovat maximální délku objektu typu string.
- **MinLength**
  Obdobně jako v předchozím případě, ale pro kontrolu minimální délky.

[1] Připojovací řetězec je serie nastavení ve formátu název=hodnota, které jsou odděleny středníkm (;) [9]

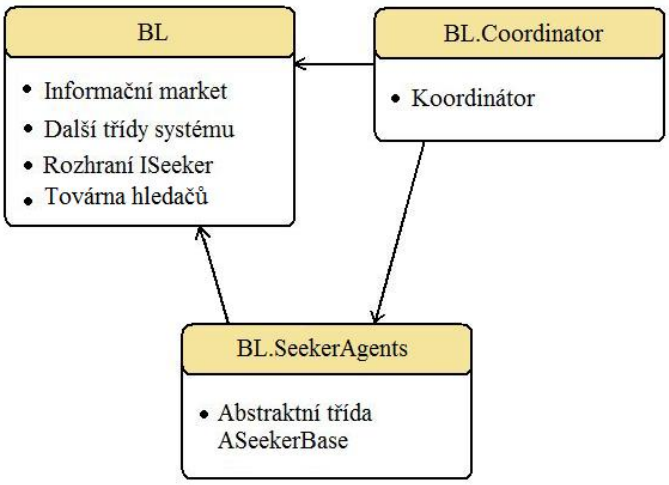
- **MaxValue**
  Tento parametr slouží ke kontrole horní hranice hodnoty typu int a double.
- **MinValue**
  Tento parametr slouží naopak ke kontrole dolní hranice hodnoty typu int a double.

6.4 Agenti

Pro implementaci agentního systému by mohlo být využito architektury popsané standardem FIPA SC00001L [14]. Vytvoření systému na základě popisu této architektury by zabralo příliš mnoho času, z toho důvodu byla pro tento systém navržena o něco jednodušší architektura. V této architektuře budou agenti reprezentováni objekty, které mají schopnost samostatné činnosti.

Pro implementaci agentního systému byly vytvořeny v rámci aplikační vrstvy 3 projekty:

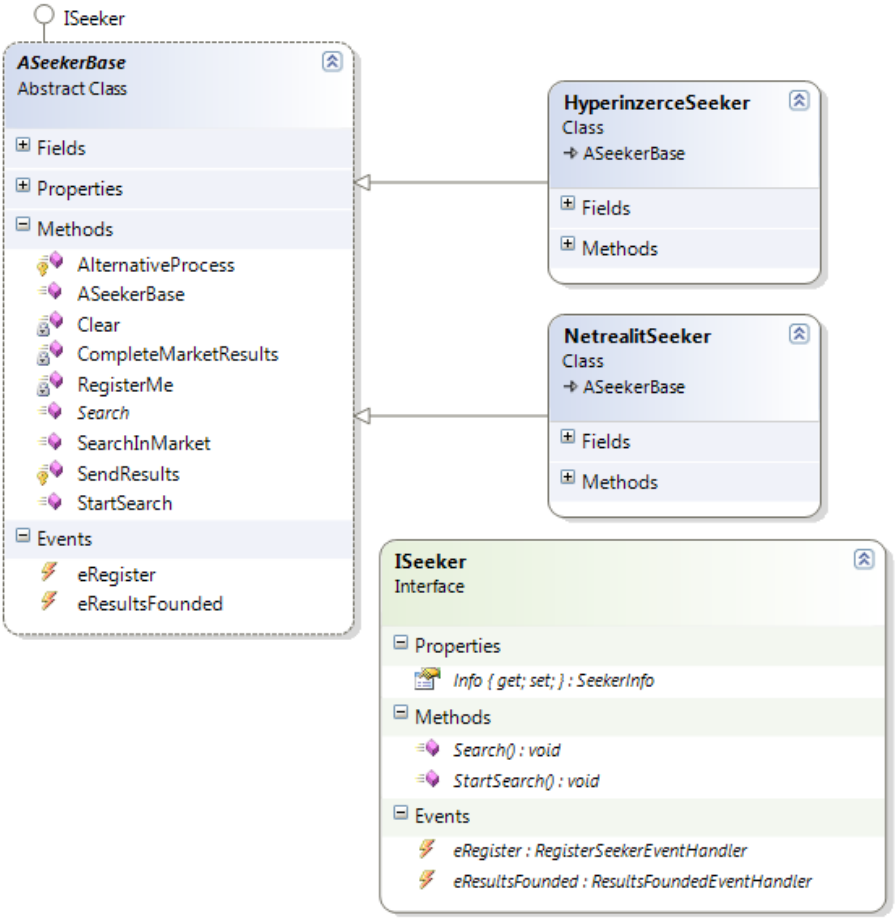
1. **BL**
   Do tohoto projektu byl zařazen informační market, rozhraní vyhledávacích agentů, továrnu pro vytváření agentů hledačů, agentní prostředí a další třídy, které nejsou již přímo spjaty s architekturou agentního systému.
2. **BL.Coordinator**
   V tomto projektu se nachazí třída agenta koordinátora.
3. **BL.SeekerAgents**
   Tento projekt obsahuje abstraktní třídu `ASeekerBase` a mohou do něj být přidávány třídy jednotlivých agentů hledačů.



Obrázek 6.1: Schéma závislostí mezi projekty

Mezi jednotlivými projekty jsou určité závislosti, které jsou znázorněny na obrázku 6.1. Závislost jednoho projektu na druhém je znázorněna šipkou, můžeme tedy ze schématu vyčíst, že projekt `BL`.Coordinator je závislý na obou dalších projektech a projekt `BL.SeekerAgents` je závislý v rámci systému pouze na projektu `BL`.

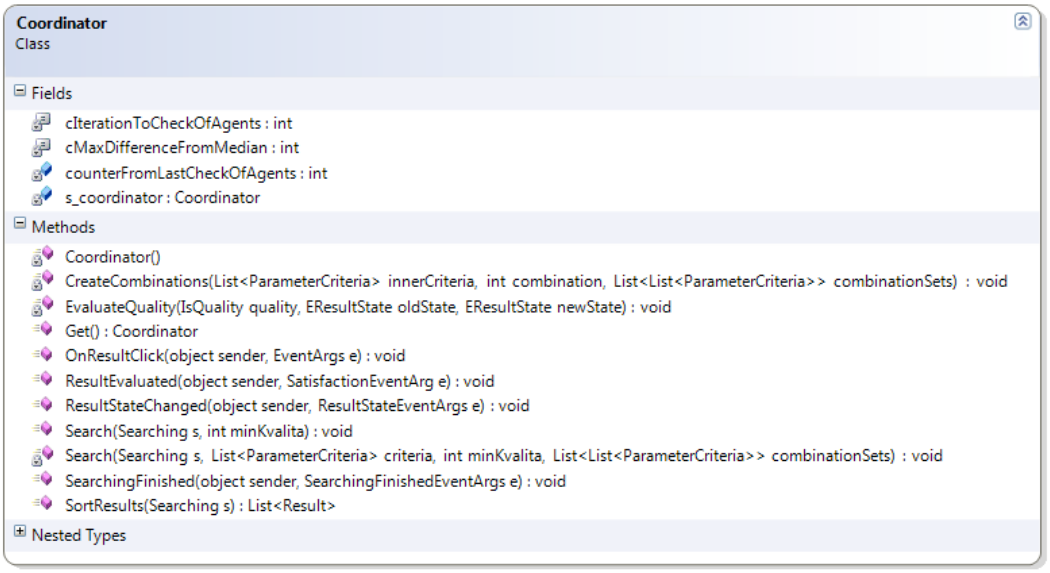
Všichni agenti hledači budou muset implementovat rozhraní `ISeeker`. Toto rozhraní předepisuje základní metody pro komunikaci s koordinátorem. Místo implementace tohoto rozhraní lze využít možnosti odvodit agenta od abstraktní třídy `ASeekerBase`, která již zajišťuje komunikaci agenta s informačním marketem a předávání nalezených výsledků koordinátorovi. Popisovaná třída obsahuje mimo jiné jednu abstraktní metodu `Search()`, která musí být implementována. V rámci implementace této metody by pak mělo být realizováno vyhledávání výsledků na konkrétním realitním serveru a přidání nalezených výsledků do výsledného seznamu. Tato třída mimo jiné nabízí také virtuální metodu `Alternative-Process()`, která je volána, pokud agent nenajde v agentním prostředí žádný jemu určený dotaz. Tato funkce je zde implementována tak, že agenta na určitou dobu uspí. Objektový model této třídy a uvedeného rozhraní je zobrazen na obrázku 6.2, kde je možné vidět i další třídy reprezentující již vytvořené agenty.



Obrázek 6.2: Hledači – objektový model

Agent koordinátor je reprezentovaný jednou třídou, která je implementována jako návrhový vzor *singleton* [11]. V systému se tedy může vyskytovat vždy jen jedna instance tohoto agenta. Koordinátor zpracovává dotazy od uživatele, které jsou mu předávány pomocí metody `Search(Searching s, int minKvalita)`. Koordinátor po obdržení dotazu na základě hodnoty předané v parametru `minKvalita` vygeneruje dotazy způsobem popsaným v kapitole 5.5, k čemuž je v této třídě implementována funkce `CreateCombinations(...)`.

Vytvořené dotazy jsou pak předány do agentního prostředí reprezentovaného třídou `AgentsEnvironment`, kde se jich ujmou jednotliví agenti hledači. Odpověď na předané dotazy pak koordinátor získá zachycením události oznamující dokončení vyhledávání. Pro její zachycení je v systému vytvořena metoda `SearchingFinished(Sender object, SearchingFinishedEventArgs e)`. Získané výsledky jsou následně ohodnoceny a do vyhledávání je nastaven příznak, který uživateli oznámí, že došlo k jeho aktualizaci. Uživatel pak získá nalezené výsledky a má možnost je hodnotit způsobem popsaným v kapitole 5.2. Ohodnocení jsou pak zaznamenávána agentem na základě zachycení událostí, k čemuž jsou určeny metody `ResultEvaluated(object sender, SatisfactionEventArgs e)`, `OnResultClick(object sender, EventArgs e)` a `ResultStateChanged(object sender, ResultStateEventArgs e)`. Objektový model této třídy je zobrazen na obrázku 6.3.



Obrázek 6.3: Koordinátor objektový model

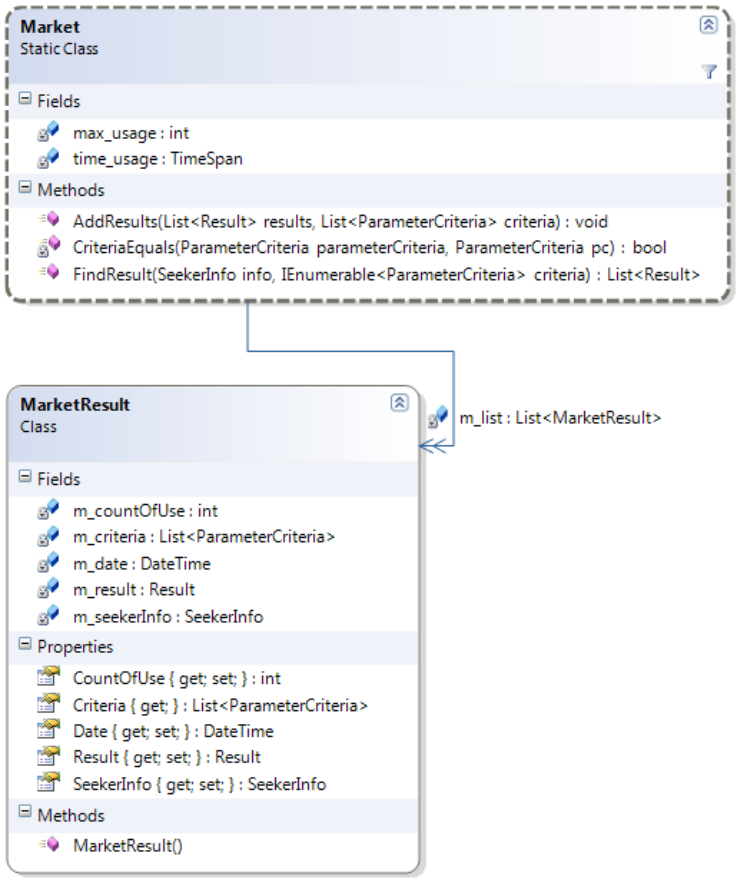
V multiagentních systémech je potřeba, aby spolu agenti navzájem nějakým způsobem komunikovali. Komunikace mezi agenty představuje klasickou výměnu informací. V tomto projektu je komunikace mezi agenty implementována dvěma způsoby. První způsob je využit při dotazování se a je reprezentován voláním konkrétní metody. Druhý způsob je použit pro odpovídání na dotaz. Odpověď je v systému předávána na základě vyvolání určité události. Komunikace mezi agenty je řešena velmi jednoduchým způsobem a využívá toho, že jsou jednotliví agenti implementováni jako samostatné třídy. Pokud by byli agenti implementováni jako samostatné programy, bylo by potřeba řešit komunikaci například na úrovni aplikační vrstvy modelu ISO/OSI. V tomto případě bychom se museli také zabývat vytvořením vlastního protokolu pro komunikaci mezi agenty, ten by pak představoval komunikační jazyk mezi agenty.

Řešení způsobu prohledávání realitních serverů, muže být řešeno velmi obdobnými způsoby. Aby bylo možné vyhnout se neustálému řešní stejných problémů dokola a byl urychlen vývoj agentů, kteří by zvládli prohledávat nové realitní servery, nabízí se možnost vytvoření agenta hledače jako univerzálního agenta. Univerzální agent by se od klasického vyhledávácího agenta moc nelišil, pouze by měl schopnost pracovat s více informacemi uloženými v databázi, a na základě informace o serveru, který má prohledávat, by

naplánoval svojí činnost. Součástí databáze by pak byla například tabulka, ve které by byly uloženy regulární výrazy pro konkrétní parametry a konkrétní realitní server. Pro rozšíření funkčnosti tohoto agenta o vyhledávání na novém realitním serveru, stačí uložit potřebné informace do tabulek v používané SQL databází.

6.5 Informační market

V systému je implemetován jednoduchý informační market postavený na free–exchange architektuře (viz. 4.2). Tento informační market je reprezentován statickou třídou, která obsahuje seznam „zboží“ `m_list` a 2 veřejné metody (`AddResults(...)`, `FindResult(...)`), pomocí kterých lze vyhledat požadované zboži a nebo přidat nové. Zboží je zde uchováváno v seznamu, kde jsou spolu s ním ukládány další dodatečné informace jako datum, kdy bylo zboží do seznamu přidáno a kolikrát bylo použito. K tomu slouží třída `MarketResult`. Seznam je průběžně kontrolován a při této kontrole je ze seznamu odstraněno zboží, které se zde nachází příliš dlouho a nebylo dlouhou dobu aktualizováno o nová kritéria. Model tohoto informačního marketu je znázorněn na obrázku 6.4



Obrázek 6.4: Informační market – objektový model

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a implementovat agentní systém pro vyhledávání na realitních serverech. Jedná se o poměrně rozsáhlý a komplikovaný projekt a z toho důvodu není jeho řešením optimalizovaný systému, který je možné začít hromadně využívat, ale pouze prototyp tohoto systému, který poskytuje inovativní přístup k vyhledávání informací na realitních serverech. Vytvořený prototyp slouží k otestování návrhu systému a je možné jej dále rozvíjet a optimalizovat ve výsledný systém. Již v průběhu testování prototypu se podařilo dosáhnout zajímavých výsledků. Agenti dokázali na základě předaných kritérií vyhledat výsledky na různých realitních serverech a následně je předat uživateli. Během krátkého času, který je na projekt vyhrazen se podařilo zprovoznit většinu zde popsaných funkcí a u zbylé části nejsou žádné zábrany, které by bránili jejímu pozdějšímu dokončení. Tento projekt se také dostal do finále soutěže Student EEICT 2010 a byl prezentován na stejnojmené konferenci.

Na záčátku této kapitoly budou uvedeny funkce systému, které nebyly implementovány v jejich plánovaném rozsahu, z důvodu jejich větší časové náročnosti. V další části je pak uvedeno zhodnocení vytvořeného systému a následně jsou uvedeny možnosti jeho dalšího vývoje a rozšiřování.

7.1 Neimplementované funkce

Jedním z cílů tohoto projektu je co nejvíce usnadnit uživatelům vyhledávání na realitních serverech. V návrhu byla uvedena řada funkcí, které by měly přispět k dosažení tohoto cíle. Je ale složité, za daného časového omezení, všechny tyto funkce implementovat v jejich plánovaném rozsahu. Zde jsou tedy uvedeny funkce, které nebyly implementovány podle popsaného plánu.

V návrhu byla popsána možnost vytváření závislých ovládacích prvků, jejichž zobrazení je podmíněno hodnotou v jiném prvku. Díky nim je možné více upřesňovat zadaná kritéria. U realitních serverů se můžeme s tímto typem parametrů setkat například při zadávání lokace hledané reality, přičemž uživatel nejdříve zadá kraj a následně je možné ještě vybrat místa, ktera se ve zvoleném kraji nachází. V implementovaném systému byl vytvořený pouze zjednodušený generátor ovládacích prvků, který zobrazuje všechny prvky a jejich možné parametry. Implementace generátoru, který by zvládal průběžně měnit generovaný obsah na základě interakce s uživatelem, by byla časově náročnější, a proto byl tento prvek systému zjednodušen. Tento fakt se pak negativně projevuje na množství informací, které musí agenti prohledat, protože jsou zpracovávány reality z rozsáhlé oblasti specifikované pouze krajem

a prozatím není možné tuto lokalitu více upřesnit. Větší množství nalezených výsledků pak vyžaduje více času, který agenti potřebují k jejich zpracování a předání uživateli.

V návrhu byla také popsána možnost vytváření různých kategorií v systému, v rámci kterých by bylo možné získávat z inzertních serverů informace i o jiném zboží než jsou nemovitosti. Implementována je ale pouze jedna kategorie a ta je zaměřená na vyhledávání bytů na realitních serverech. Pro kompletní prohledávání realitních serverů by měly být vytvořeny další kategorie, které by se specializovaly na vyhledávání domů, pozemků, kanceláří a dalších nemovitostí, které do této oblasti přísluší. Další kategorie, které by mohl tento systém poskytovat, jsou již jen rozšířením této práce, a proto zde také nebyly vytvořeny.

7.2 Zhodnocení

Implementovaný systém obsahuje dva různé vyhledávací agenty. První z těchto agentů byl vytvořen pro základní otestování celého systému a slouží k získávání informací ze serveru `www.hyperinzerce.cz`. Druhý agent byl pak vytvořen pro důkladnější otestování komunikace mezi agenty a jeho úkolem je vyhledávat informace na serveru `www.netrealit.cz`.

Vytvoření agenti jsou implementováni tak, že poskytují výsledky, které jsou stejně relevantní jako výsledky, které by uživatel získal s použitím samotného realitního serveru. Jejich získávání ale prozatím vyžaduje relativně dost času, k jehož zkrácení sice dochází postupně díky využití informačního marketu, ale bylo by potřeba tento proces ještě více zrychlit.

Systém prošel jen jeho základním otestováním. Pro jeho důkladnější otestování by bylo zapotřebí zpřístupnit jej širší veřejnosti, aby bylo možné lépe nastavit implementované ohodnocovací funkce. K tomu je ale zapotřebí získat informace od uživatelů o spokojenosti s nalezenými výsledky. Toto testování by ale vyžadovalo určité finanční náklady, které by byly v této fázi zbytečné, kvůli nedostatečné optimalizaci tohoto systému.

Slabinou tohoto systému je tedy nedostatečná optimalizace. Řešení tohoto problému budou shrnuta v následující části práce, která se zabývá dalšími možnostmi vývoje. Systém ale poskytuje inovativní přístup k vyhledávání, který zatím jiné systémy nenabízí a který by mohli jeho uživatelé ocenit. Mezi hlavní přednosti systému pak patří to, že se snaží uživateli nalezené výsledky co nejvíce zpřehlednit, tak aby se v nich při jejich opětovném procházení lépe orientoval.

7.3 Další možnosti vývoje systému

V předchozí části bylo uvedeno, že vytvořený systém představuje pouze prototyp navrženého systému. Je potřeba se tedy zabývat jeho dalším vývojem a to jak optimalizací již vytvořených částí, tak vytvářením dalších funkcí, které by tento systém mohl poskytovat.

Jelikož slabinou systému je nižší rychlost, další vývoj systému by měl být zaměřen na optimalizaci, ta by se měla zaměřit na nejvíce využívané části tohoto systému, kterými je datová vrstva systému a činnost agentů. V rámci datové vrstvy je potřeba vylepšit práci s agregovanými objekty, které se momentálně musí neustále znovu načítát z databáze, ikdyž nedošlo k jejich změně. Řešením tohoto problému by mohlo být vytvoření speciální kolekce, která by sloužila jako přístupový bod k agregovaným objektů a která by umožnila redukování potřebného načítání dat z databáze.

Agenti hledači musí mnohdy získávat velké množství informací a jejich získávání vyžaduje určitou síťovou komunikaci. Při této komunikaci dochází k určité časové prodlevě, a proto by

měla být do budoucna redukována. Díky tomu by došlo ke snížení vytížení tohoto systému, ale i samotných realitních serverů. K redukování této komunikace by mohlo přispět průběžné získávání výsledků k jednotlivým parametrům, které by byly ukládány v informačním marketu. Uživateli by pak byly předány pouze informace nalezené v tomto informačním marketu a nemusel by čekat než proběhne celý proces získávání informací z realitních serverů. Informační market by sice nemusel poskytovat všechny informace, které je možné získat, ale výsledky by zde byly postupně přidávány a aktualizovány na základě dotazů uživatelů.

Systém by mohl být dále rozšířen o možnost shlukování stejných inzerátů nalezených na různých realitních serverech. K této ale i předchozí možnosti rozšíření by bylo zapotřebí získávat konkrétní hodnoty pro parametry, jejichž vstupem je předem neznámá hodnota. Pomocí tohoto parametru je například zadávána minimální a maximální cena.

Možností dalšího rozšíření systému jsou ještě obrovské. Určitě mezi ně budou patřit funkce, které se zde kvůli nedostatku času nepodařilo implementovat, ale také další funkce, mezi které může patřit například automatické upravování vah ohodnocovacích funkcí prováděné koordinátorem, zobrazování více detailů o nalezených výsledcích aj.

Literatura

[1] Agarwal, A. S.: SoftBotSearch : SoftBot Approach to information retrieval from Web [online]. http://www.hipc.org/hipc2005/posters/shubham.pdf, 2005-9-12 [cit. 2009-12-28].

[2] Altman, A.; Moshe, T.: Ranking Systems: The PageRank Axioms [online]. http://stanford.edu/~epsalon/pagerank.pdf, 2005-5-31 [cit. 2010-2-28].

[3] Bartsch, H.-J.: *Matematické vzorce*. Academia, 2006, ISBN 80-200-1448-9, 832 s.

[4] Battelle, J.: *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Nicholas Brearley Publishing, 2005, ISBN 1-85788-361-6, 311 s.

[5] Dragut, E.; Fang, F.; Sistla, P.; aj.: Stop Word and Related Problems in Web Interface Integration [online]. http://www.dit.unitn.it/~p2p/RelatedWork/Matching/vldb09a.pdf, 2009-8-28 [cit. 2009-12-28].

[6] Dudka, K.; Durman, D.; Filák, J.; aj.: Softbot pro získávání informace z internetu [online]. http://www.fit.vutbr.cz/~ifilak/pub/softbot.pdf, 2008 [cit. 2009-12-28].

[7] Graat, G.: *Agent-based Information Retrieval Supported by Information Markets [online]*. Diplomová práce, Faculty of General Sciences of the Universiteit Maastricht, 2003-2-28 [cit. 2010-03-20].

[8] Šimara, M.: Agent System for Searching on Real Estate Servers. In *Proceedings of the* 16^{th} *conference Student EEICT*, ročník 1, editace M. Drahanský; F. Orság, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií a Fakulta informačních technologií, Brno: NOVPRESS s.r.o., 2010, ISBN 978-80-314-4076-0, s. 207–209.

[9] MacDonald, M.; Szpuszta, M.: *ASP.NET 3.5 C# 2008: tvorba dynamických stránek profesionálně*. Zoner Press, druhé vydání, 2010, ISBN 978-80-7413-008-3, 1584 s.

[10] Odubiyi, J. B.; Kocur, D. J.; Weinstein, S. M.; aj.: SAIRE - A Scalable Agent-based Information Retrieval Engine [online]. http://sigart.acm.org/proceedings/agents97/A023/A023.PDF, 2008 [cit. 2010-3-12].

[11] Pecinovský, R.: *Návrhové vzory*. Brno: Computer Press, 2008, ISBN 978-80-251-1582-4, 527 s.

[12] Ramos, J.: Using TF-IDF to Determine Word Relevance in Document Queries [online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf, [cit. 2010-3-19].

[13] Zbořil, F.: *Plánování a komunikace v multiagentních systémech [online]*. Dizertační práce, FIT VUT v Brně, 2004 [cit. 2010-03-19].

[14] Zbořil, F.: Agentní a multiagentní systémy - Studijní opora (modul 1) [online]. http://perchta.fit.vutbr.cz:8000/vyuka-ags/uploads/3/AGS_opora_m1_ESF.pdf, 2006 [cit. 2009-12-28].

Příloha A

Obsah CD

- Adresář *projekt*
  V tomto adresáři se nacházejí 2 projekty (PVObject a AS_WebApplication), které jsou produktem této práce. Dale je zde projekt (ASP.NET AJAX Library), který slouží k vytvoření knihoven, které jsou potřebné při kompilaci aplikace. Obsahem je také soubor readme.txt, ve terém jsou popsány potřebné úpravy pro úspěšnou kompilací a úspěšné spuštění projektu.

- Adresář *text*
  Zde se nachází tato práce ve formátu PDF.

- Adresář *latex*
  Tento adresář obsahuje zdrojové kódy pro systém LaTeXa obrázky, pomocí kterých je možné vygenerovat tento dokument.

Příloha B

Manual

Jednotlivé kroky pro použití:

1. Registrace – Pokud jste již v systému zaregistrováni, tento krok přeskočte a přejděte k příhlášování. Na úvodní straně jsou zobrazeny dva formůláře jeden slouží k přihlášení do systému a druhý k registraci. Pro registraci je potřeba zadat login a heslo. V tomto formuláři je možné vyplnit i emailovou adresu, která může sloužit i jako login při přihlašování. Po vyplnění těchto údajů je potřeba potvrdit zadané údaje stiskem tlačítka s nápisem „Registrovat“. Systém pak registraci potvrdí a nebo oznámí proč se registrace nezdařila.

2. Přihlášení – pokud již máte své registrační údaje, použijte je ve formuláři pro přihlášení do systému. Po přihlášení budete přesunuti na záložku „Hledání“. Nyní máte možnost pracovat s vámi dříve vytvořenými vyhledáváními, vytvořit nové vyhledávání nebo se ze systému odhlásit.

3. Nové vyhledávání – V levém menu najdete položku s názvem „Nové vyhledávání“. Po kliknutí na tuto položku se zobrazí formulář, na kterém jsou zobrazeny dva prvky. Pomocí prvního je potřeba zadat název vyhledávání a pomocí druhého vybrat určitou kategorii. Po vybrání této kategorie se na formuláři zobrazí seznam dalších prvků, které slouží jako parametry pro vyhledávání. Zde nastavíte vámi požadované kritéria pro vyhledávání a po kliknutí na název libovolného parametru můžete nastavit povinnost tohoto parametru při případném dohledávání většího množství výsledků. Parametr označený jako povinný je po chvíli zvýrazněn žlutým podbarvením. Po vyplnění všech informací je potřeba stisknout tlačítko „Přidat“, což způsobí vytvoření nastaveného vyhledávání. Následně máte možnost pokračovat ve vytváření dalších vyhledávání a nebo pomocí položky s názvem „Moje vyhledávání“ přejit k práci s vytvořenými vyhledávaními.

4. Aktualizace vyhledávání – Pokud jste vytvořili nové vyhledávání, je potřeba ho nejdříve aktualizovat aby se provedlo vyhledávání a vám byly zobrazeny nalezené výsledky. K tomu je potřeba pomocí výběru kategorie a vyhledávání v horní části formuláře najít a vybrat konkrétní vyhledávání, a poté kliknout na tlačítko „Aktualizovat“, které se nachází pod menu na levé straně od formuláře. Tuto aktualizaci pak můžete opakovat pokaždé, když budete chtít zkusit vyhledat nové výsledky. Po dokončení aktualizace jsou výsledky zobrazeny ve spodní části formuláře.

5. Nastavení filtru – Na formuláři se nachází jednoduchý filtr, pomocí kterého lze nastavit, aby se zobrazovaly jen výsledky patřící do určitých kategorií. Dále umožňuje nastavit přesnost výsledků, která určuje jak moc musí být splněny všechny kritéria. Změnu nastavení filtru je pak potřeba potvrdit a v případě snížení přesnosti provést aktualizaci, pokud již nebyla pro nižší nebo stejnou přesnost provedena dříve.

6. Hodnocení výsledků – Nalezené vysledky lze hodnotit třemi způsoby, které byly popsány v kapitole 5.2. Tyto možnosti jsou zobrazeny na obrázku B.1. V jeho levé části je vyznačeno nepřímé hodnocení výsledku vyvolané změnou kategorie. Panel s možností volby se zobrazí, pokud do vyznačeného místa najedete ukazatelem myši. Další možností, kterou je hodnocen výsledek, je prohlížení si výsledku na realitním serveru. To je umožněno kliknutím na název výsledku, což je vyznačeno uprostřed obrázku. Poslední možností je přímé hodnocení výsledku pomocí prvku vyznačeného na pravé straně obrázku. Tento prvek po vašem hodnocení zmizí.

Obrázek B.1: Hodnocení nalezených výsledků

7. Smazání vyhledávání – V závěru, když jste již našli požadované informace, můžete vytvořené vyhledávání smazat pomocí tlačítka, které se nachází na stejném místě jako tlačítko „Aktualizovat“.